期末项目实验报告

1. 项目介绍以及实现结果

1.1 项目介绍

本项目实现了一个第一人称户外射击游戏，玩家可以进行移动人物，举枪开镜射击等一系列的操作，通过射击靶子得分。

1.2 实现结果

本项目实现了玩家通过键盘输入控制人物移动，通过移动鼠标控制人物视角的旋转。同时，本项目还实现了右键开镜和左键射击的操作。场景上本项目使用了大量的模型，实现了粒子系统等高级物理效果，而且进行了抗锯齿，实例化数组等画面优化设计。在物理系统方面实现了重力系统和碰撞系统，分别用于模拟玩家的跳跃以及子弹命中的判定。

2. 开发环境以及使用到的第三方库

2.1 开发环境

Windows 10，Visual Studio 2017，OpenGL

3.1 第三方库

ImGUI 库，FreeType 库，Assimp 库，stb_image 库

3. 实现功能列表

3.1 Basic 功能

(1) Camera Roaming（摄像机漫游）
(2) Simple lighting and shading（光照）
(3) Texture maping（纹理贴图）
(4) Model import and Mesh viewing（模型加载）

3.2 Bonus 功能

(1) 天空盒
(2) 粒子系统
(3) 抗锯齿
(4) 实例化数组
(5) 法线贴图
(6) 文字显示
(7) 重力系统与碰撞检测
(8) 爆炸效果
(9) Gamma 校正
(10) 面剔除

4. 实现功能点介绍

（1）Basic 部分

知识点：Camera Roaming（摄像机漫游）

说明：摄像机的漫游主要是构建 View 矩阵，将世界坐标系中的物体转换到摄像机坐标系中。这部分通过接受用户输入，实时改变摄像机的前向量，右向量，动态修改 View 矩阵，实现视角的移动。

知识点：Simple lighting and shading（光照）

说明：使用了 Phong 光照模型，以及使用 Blinn-Phong 改进高光效果。

知识点：Texture maping（纹理贴图）

说明：模型贴图以及法线贴图等都使用了纹理贴图的功能。

知识点：Model import and Mesh viewing（模型加载）

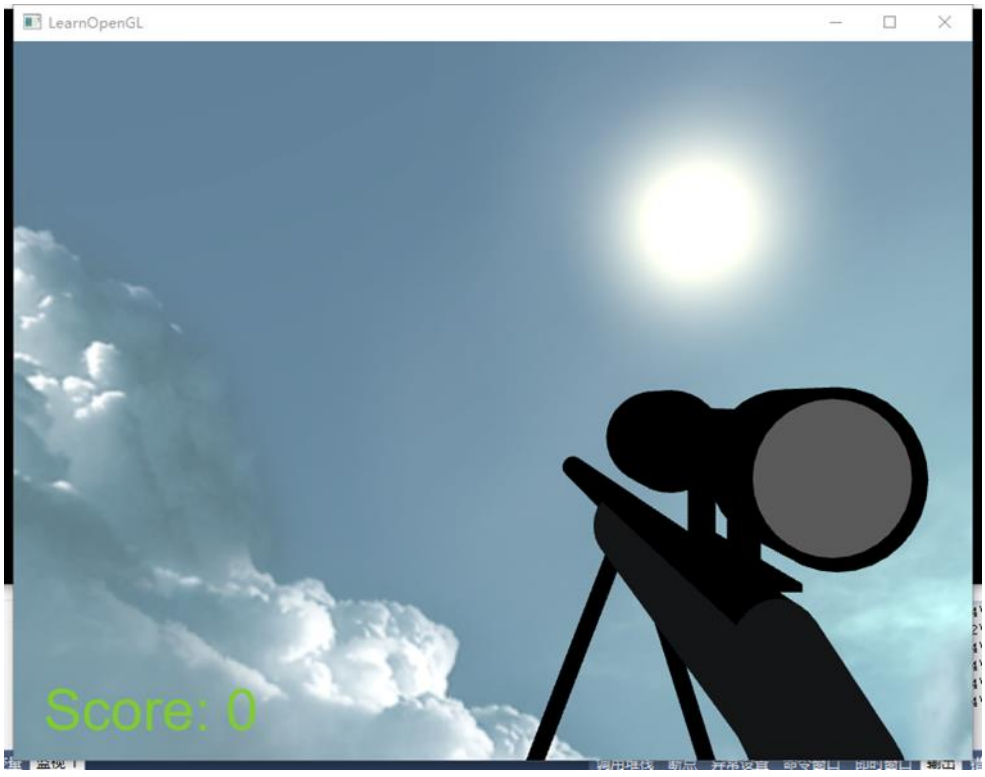
说明：这部分通过使用 Assimp 库加载 OBJ 模型实现。

（2）Bonus 功能

知识点：天空盒

说明：天空盒使用一个立方体贴图盒,将摄像机的 view 和 projection 矩阵传进着色器的时候只需要将位移项去除即可。

演示截图:(天空盒的代码封装在 skybox.h 文件)



知识点：粒子系统

说明：首先我们实现了一个粒子系统，又一个粒子对象和一个粒子发射器构成，粒子发射器会不断产生粒子，粒子的属性如速度、颜色、透明度等，在整个生命周期中逐渐变化直到消亡，从而产生各种奇妙的特效，我们选择用粒子系统模拟的是火焰:

- 火焰粒子使用的是一个烟雾状的材质贴图
- 颜色模拟：火焰粒子生命周期颜色由白到黄到红的渐变
- 形状：随机在火焰中心区域生成粒子，速度由一个背向中心的向量和一个向上的向量以一定比例组成。
- 随机旋转粒子贴图，使得火焰效果可以立体化观测。

演示截图:(粒子系统的代码封装在 particle.h 文件)

知识点：抗锯齿

说明：抗锯齿的实现分为两部分。

- 对整体图像的抗锯齿优化，使用了 OpenGL 内建的多重采样抗锯齿（MSAA）算法。使用抗锯齿优化之前，由于在光栅化的阶段，每个像素点只包含一个中心采样点。在屏幕像素总量的限制下，会出现明显的走样现象。多重采样技术将单一的采样点变为多个（这里是 4 个）采样点，通过平均化多个采样点的颜色值，得到更平滑的图形。
- 阴影抗锯齿，使用了 PCF（percentage-closer filtering）方法。因为阴影贴图解析度的限制，在进行阴影映射时，多个片元可能对应同一个阴影纹理像素，导致多个片元得到同一个深度值，因此会产生阴影锯齿。PCF 也是一种多次采样的思想，在阴影映射时多次移动阴影贴图，对深度贴图多次采样，最后将亮度值平均，得到更加平滑的阴影。

演示截图：

- 整体图像抗锯齿（左为优化前，右为优化后）
  代码：

```
glfwWindowHint(GLFW_SAMPLES, 4);
glEnable(GL_MULTISAMPLE);
```


- 阴影抗锯齿（左为优化前，右为优化后），代码在 bin/ShaderCode/3.phong_shading.fs 中。


知识点：实例化数组

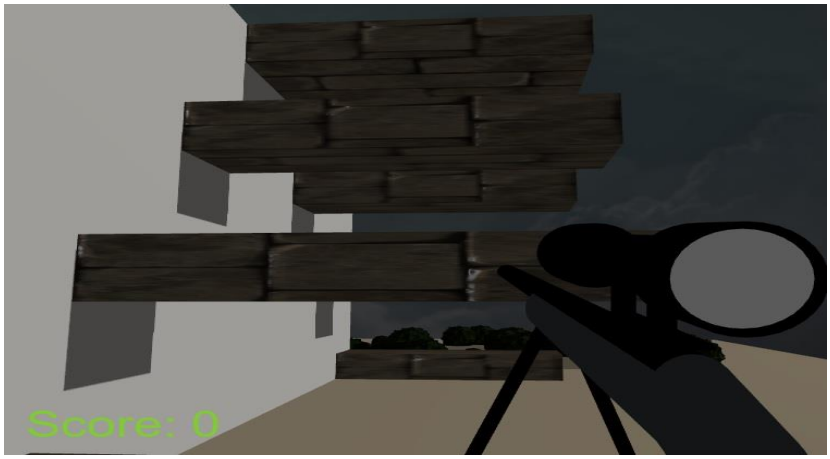
说明：使用实例化数组渲染能大大加快模型的渲染速度，减少运行时的卡顿。这种方式特别适合渲染大量重复的模型，例如我们的草和树木。首先，我们要在顶点着色器中增加一个 mat4 顶点属性，存储一个实例化数组中的变换矩阵，代替原有的全局变量 model，以实现世界坐标的变化。然后，我们将包含多个变换矩阵的实例化数组绑定到一个顶点缓冲对象中。接下来，设置顶点属性指针，并用 glVertexAttribDivisor 将顶点属性设置为实例化数组，以便能够将变换矩阵一次性传给着色器。最后，使用 glDrawElementsInstanced 绘制特定数量的实例。这样做减少了 CPU 和 GPU 通信的次数，减轻了总线瓶颈对性能的影响。

代码：着色器代码见 bin/ShaderCode/instancing_depth_mapping.vs 和 bin/ShaderCode/instancing_phong_shading.vs。实例化数组渲染过程见 src/GameTools.h。

知识点：法线贴图

说明：法线贴图不使用插值表面法线，而是为每个 fragment 传递一个法线，可以使光照使表面拥有了自己的细节。首先将法线向量变换为 RGB 颜色元素，我们就能把根据表面的形状将 fragment 的法线保存在 2D 纹理中，由于法线贴图里面的所有法线向量都是指向正 z 方向的，所以引入切线空间来解决面朝向其他方向时的问题，法线贴图中的法线向量在切线空间中永远指着正 z 方向，使用一个特定的矩阵我们就能将本地/切线空间中的法线向量转成世界或视图坐标，使它们转向到最终的贴图表面的方向。然后再使用该 2D 纹理来加载法线即可实现法线贴图。

演示截图：



知识点：文字显示

说明：这部分使用 FreeType 库实现。FreeType 库从系统导入字体库，由于我们只实现了英文字符的输入，因此可以只为 128 个 ASCII 码的每个码绑定一张对应的贴图，渲染时通过开启 Blend 使之能够与背景融合。

知识点：重力系统与碰撞检测

说明：

（1）重力系统的实现是通过使用物理公式计算出玩家视角在垂直方向上的速度以及在这段时间内的位移，进而调整玩家视角位置模拟跳跃动作。而在计算位移时，我们采用 $S = S0+v\Delta t$ 这条公式，而不使用 $S=S0-gt^2/2$ 这条公式。这是因为 OpenGL 是每隔Δt 刷新屏幕，而且时间极短。于是，我们可以将玩家在刷新的这段时间内的运动轨迹当作是匀速直线运动，从而简便了计算过程。

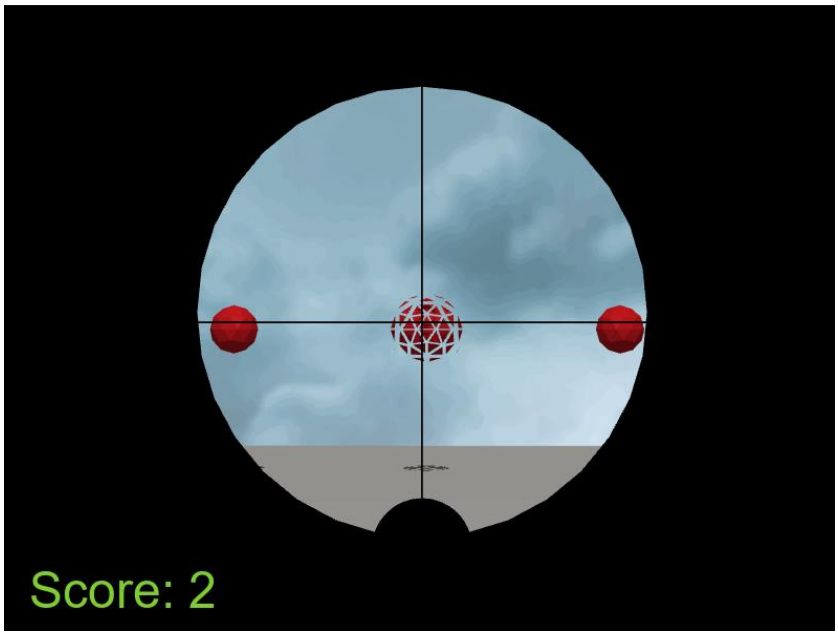
（2）碰撞检测的实现主要采用分成两种形式：场景外边界的碰撞，场景内物体的碰撞。场景外边界的碰撞很简单，只需要检测玩家在 XOZ 平面内的坐标是否超出预设的场景边缘即可；而对于场景内物体的碰撞，就比较复杂。首先需要为每个物体设置一个碰撞盒，然后再做碰撞检测时，首先检测玩家和物体在 XOZ 平面内的投影是否出现了碰撞（AABB 碰撞）。若 XOZ 平面出现了碰撞，如果此时玩家的高度（Y 分量）低于物体的最低高度或高于物体最高高度，那么认为玩家没有碰撞到物体，否则认为出现了碰撞，执行碰撞处理函数。

代码：重力系统与碰撞检测系统的代码封装在 PhysicEngine.h 文件

知识点：爆炸效果

说明：通过几何着色器实现。在几何着色器内，首先通过一个面的三个顶点坐标计算出当前面的法线向量，然后将这三个顶点同时沿法线向量偏移一段距离，最后将偏移后的顶点坐标向量作为新的坐标向量输出。

演示截图:



Gamma 校正

说明：这部分主要是对图像进行色彩增强和色彩还原。Gamma 校正对图像的 Gamma 曲线进行编辑，以对图像进行非线性色调编辑的方法，检出图像信号中的深色部分和浅色部分，并使两者比例增大，从而提高图像对比度效果。一般显示器使用 sRGB 颜色空间，此时 gamma 值一般为 2.2。

Gamma 校正还有一个附加内容是增加光照衰减效果。按照物理世界的规律，物体离光源越远，其受到的光照强度越低。因此我们可以在环境光和高光系数前乘以一个衰减系数以模拟真实光照效果。

演示截图:

Gamma 校正前

Gamma 校正后

知识点：面剔除

说明：由于摄像机在观察一个立方体时，最多只能同时观察到其中的三个面。因此我们可以在渲染的过程中只渲染看得到的面（正向面），抛弃无法观察到的面（背向面），从而节省着色器资源，提高渲染速度。而我们在定义模型的每个面时，一般使用逆时针的顺序定义其顶点。这种定义方法使得从观察者的视角看，正面的三角形以逆时针的顺序渲染，而背面的三角形以顺时针的顺序渲染。根据这种性质，我们可以通过定义正向面的渲染顺序，控制 OpenGL 剔除面的类型。

5. 遇到的问题和解决方案

问题	解决方案
实现爆炸效果时阴影贴图无法正确显示爆炸物体的阴影效果	为深度着色器添加同样的几何着色器，在几何着色器内先计算出偏移后的顶点坐标向量，然后再将新坐标向量映射到光空间。
火焰粒子的需要相互混合与场景物体之间的需要相互遮盖冲突的问题	颜色混合跟深度测试其实是存在一定对立关系的，要使场景能呈现自然的效果需要小心设置它们的在渲染流程中的开关以及条件设置。

6. 小组成员的分工

学号	成员	分工	比重
16340089	黄铸韬	重力系统与碰撞检测；文字渲染；Gamma 校正；模型加载；多重采样抗锯齿；面剔除	26%
16340108	黎浩良	天空盒的实现与布置；全局光照阴影效果，以及相应的改进；昼夜场景模拟；粒子系统的实现以及粒子模拟火焰的效果	27%
16340109	黎汛言	基础 Phong 光照；阴影映射；实例化数组渲染；场	24%

		景模型布置	
16340094	江炎鸿	场景中树、草、枪械、石头等模型的设计、制作和渲染；法线贴图	23%

7. 个人报告

黄铸韬

这次课程作业我负责整个项目框架的搭建。在这个项目中，我也负责了一些比较重要的模块的实现。一是实现了比较有趣的重力系统与碰撞检测系统，能够较好的模拟真实人物跳跃的视角变换。二是尝试了课堂上没用过的几何着色器对模型形态进行变换，对着色器以及 OpenGL 的渲染管线有了更深刻的了解。三是尝试了多种渲染优化方式，如 Gamma 校正，抗锯齿，面剔除等，使得整个项目的场景的渲染速度得到提升，视觉效果更加流畅。

黎浩良

这次的课程作业我负责的功能点之中比较重要的有天空盒与昼夜模拟、全局光照阴影以及改进、粒子系统模拟火焰等。这些功能中是存在不少难点的，我也花费了相当的心思和时间去设计、实现和测试。在做天空盒时，为了整个场景的昼夜场景与天空盒的效果契合，设置了一个亮度衰变的系数，在光源围绕整个场景旋转时同时变化。在做全局光照时，要设计一个可以容纳复杂场景中全局光照的架构，需要对渲染流程十分了解以及充分细心，否则会遇到很多坑，比如光源的前向量不能跟摄像机的世界变量共线，以及加载模型时绑定的纹理贴图容易与阴影贴图出现冲突等问题。在做粒子系统时遇到的难点最多，除了设计一个比较粒子系统的架构，还要制作合适的粒子材质，构思如何模拟火焰的颜色、形状、速度和立体感等，而且为了达到更好的模拟效果，对深度测试的调用和颜色混合的设置都至关重要。总的来说，这次期末项目不仅复习了课堂知识，也让我学习扩展了很多图形学相关的知识。

黎汛言

这次课程作业我主要负责了基础 Phong 光照、阴影映射、场景模型布置这些基础内容，也完成了场景抗锯齿、实例化数组渲染等优化工作。基础光照实现起来不是很难，基本思路与曾经做过的作业相似。在做阴影映射时，我完成了阴影贴图的基础代码框架，但是实际运行起来存在不少问题，例如靶子的阴影没有正确显示。在此要感谢黎浩良同学在阴影映射方面的后期优化。在测试过程中，我发现模型数量的增加会严重影响程序性能，因此尝试了实例化数组的渲染方式，获得了不错的效果。不过刚开始修改代码时没有正确绑定好贴图，导致模型的颜色出现异常，在此感谢黄铸韬同学对该问题的指出。通过这个项目，我对图形学课程中学到的知识点进行了一次巩固，也学到了不少课堂上没有提到的技术，对计算机图形的基本处理过程有了更清晰的理解。

江炎鸿

在这次的课程作业中我主要负责的是场景中树、石头、草、枪等模型的设计、制作和渲染以及法线贴图部分。对于场景中的树、草、枪械等模型，一开始是打算上网寻找现有的模型，但是找到的模型不知道为什么都无法加载到项目中，所以最后我只能自己利用 Blender 进行制作，然后导出为 obj 文件使用，为了使场景不要太单调，树跟石头我都做了三种样式。至于法线贴图的部分，教学网站上讲得比较详细，所以比较好理解，在实现的时候比较麻烦的就是要跟队友写的 Gamma 矫正、光照阴影等进行合并。